Appendix I

This appendix describes a number of answer generators, starting with one that can provide answers to natural-language questions that are grammatically context-free, and then to those for other types of questions. Different tasks in the following description performed by different elements can be implemented by the interaction controller.

A natural-language question can be in English or other languages, such as French. Examples of natural-language questions are:

Who is the first President?

What are the Bills of Right?

Where is the capital of Texas?

A statement that is not based on a natural language is a statement that is not commonly used in our everyday language. Examples are:

For Key in Key-Of(Table) do

Do while x>2

A grammatically-context-free question is a question whose grammar does not depend on the context. Each word in the question has its own grammatical meaning, and does not need other words to define its grammatical meaning. Hence, the grammatical structure of the question does not depend on its context.

The question includes one or more grammatical components. A grammatical component is a component with one or more grammatical meanings, which are defined by a set of grammatical rules to be explained below. For example, the word "president" is a noun, which has a grammatical meaning. So the word "president" is a grammatical component.

In one embodiment, the question-answering approach includes a database with a number of tables. The data in each table can be further divided into different areas, and each area is represented by an attribute. Some values or data in the database may be unique. Such values are known as key values, and their corresponding attributes are known as key attributes.

One embodiment of the database includes a grammatical table, one or more topic-related tables, and two semantic tables. In a general sense, the grammatical table determines

the grammatical meaning of each word in the question, such as whether a word is a noun or a verb. Each topic-related table groups data related to a topic together in a specific format. Separated into a topic-dependent semantic table and a topic-independent semantic table, the semantic tables define the semantic meaning of each word, such as whether a word refers to an algorithm or data in a topic-related table.

The grammatical table defines the grammatical meanings of words used in the natural-language question. If questions entered into the system is limited to only one subject, such as history, the grammatical table will include words in that subject, and words commonly-used by a user of the system in asking questions.

Each topic-related table combines data related to a topic in a specific format.

There is also a table-structure dictionary, which defines how the topic-related tables arrange their data. This dictionary is typically not considered as a part of the database. It does not contain topic-related data, but it contains structures of the topic-related tables in the database.

A word in the question may need one or both of the semantic tables. The topic-independent semantic table defines whether a word stands for an algorithm or data in a topic-related table. Such a table may be defined as follows:

```
CREATE TABLE Topic_Independent_Semantic (
        word            NOT NULL,   // the word
        semantics,                  // Indicates if the word refers to data in a
                                    // topic-related table, an algorithm etc. If the
                                    // word is mapped to an algorithm, that
                                    // algorithm will also be identified, as will be
                                    // further explained below.
        synonym,                    // A word might have synonyms, as will be
                                    // further explained below.
)
```

Words with similar meaning are grouped together and are represented by one of those words as the synonym for that group of words.

Many words do not point to an algorithm. They correspond to data in topic-related tables. The topic-dependent semantic table identifies the semantic meaning of those words through matching them to data in topic-related tables. Such a topic-dependent table may be defined as follows:

```
CREATE TABLE Topic_Dependent_Semantic (
  Table_Name NOT NULL, // For a table with the name Table_Name:
    Who_Attribute,        // The attribute associated with 'who'
    When_Attribute,       // The attribute name associated with 'when'
    {i-pronoun}_Attribute, // The attribute associated with an
                          // interrogative pronoun or i-pronoun.
                          // The symbols {} denote the word it
                          // contains. Here, the word is an i-pronoun.
    ...
    {Adj}_Attribute,
      // The attribute associated with the adjective {adj}. In this
      // example, the word is an adjective.
    {Noun}_Attribute,
      // Attribute name associated with the noun {noun}. Certain
      // nouns may refer instead to an algorithm, such as "sum."
)
```

In general terms, a grammatical structure analyzer can analyze the grammatical structure of a natural-language question so as to parse it into its grammatical components, based on a pre-defined context-free grammatical structure. This task uses a set of grammatical rules and the grammatical table. Then, the system transforms at least one component into one or more instructions using a set of semantic rules with one or both of the semantic tables. Finally, one or more steps are executed to access and process data from one or more topic-related tables so as to generate an answer to the question.

Analyze Grammatical Structure

In one embodiment, the analyzer scans the question to extract each word in the question. Then the analyzer maps each extracted word to the grammatical table for identifying its grammatical meaning. After establishing the grammatical meaning of each word, the analyzer uses a set of grammatical rules to establish the grammatical components of the question based on a pre-defined context-free grammatical structure.

In one embodiment, the pre-defined context-free grammatical structure is as follows:

<Question> = <i-pronoun> <aux-verb> <noun-phrase> [<verb-phrase>]

where: the symbols <> denote whatever inside is a meta-symbol, which has a grammatical meaning; the meta-symbol is not in the grammatical table.

The symbols [ ] denote whatever inside the bracket is optional.

<I-pronoun> denotes an interrogative pronoun, which is a pronoun used in asking questions, and can be one of the following: what, when, where, who, whom, whose, which, and why.

<Aux-verb> denotes an auxiliary verb, and can be any form of the verb "to be," or "do."

<Noun-phrase> is defined as <group-of-nouns> [<prepositional-noun-phrase>]

where: <group-of-nouns> is defined as:

[<modify-article>] <adjective>* <one-or-more-nouns>;

the symbol * denotes zero or more;

<modify-article> is defined as a modified article, including a, an, the, this, these and those; and

<one-or-more-nouns> denotes one or more nouns; and

<prepositional-noun-phrase> is defined as a

<preposition> <noun-phrase>.

<Verb-phrase> denotes a non-aux-verb, and

is defined as <non-aux-verb> [< prepositional-noun-phrase>].

<Preposition> denotes a preposition defined in the grammatical table.

<Non-aux-verb> denotes a verb defined in the

grammatical table and is not an <aux-verb>.

<Noun> denotes a noun defined in the grammatical table.

<Adjective> denotes an adjective defined in the grammatical table.

A word or a set of words that can fit into the structure of a meta-symbol is a grammatical component. For example, the phrase "with respect to x" is a grammatical component, whose grammatical meaning is a prepositional-noun-phrase.

The grammatical table defines the grammatical meaning of each word.

Many questions cannot be parsed based on the pre-defined context-free grammatical structure. These questions are considered as ambiguous questions, and will be analyzed through methods explained later.

Programming-steps generator

The programming-steps generator transforms at least one grammatical component of the question using a set of semantic rules and one or both of the semantic table to generate a set of instructions. The semantic rules and the semantic tables depend on the pre-defined context-free grammatical structure, which the parsing process bases on.

To help explain question-answering approaches, a number of functions are created as shown in the following:

- Keys-Of(Table)

  This function extracts all the key attributes in the identified table.
- Attributes-Of(Table)

  This function extracts all the attribute names in the identified table.
- Attribute-Names({adjective}, Table)

  This function identifies one or more attributes when the {adjective} is applied to the table.
- Attribute-Names({noun}, Table)

  This function identifies one or more attributes when the {noun} is applied to the table.
- Attribute-Name({i-pronoun}, Table)

  This function identifies the attribute when the {i-pronoun} is applied to the table.

- Tables-Of({proper noun})

  This function identifies one or more tables that contain the {proper noun} as a key value. It can be derived by the following program:

```
T-Names = "";
for Table in {all Tables}   // {all Tables} is a list of topic-related tables
do
    for Key in Keys-Of(Table)
    do
        if any value of the attribute Key in the Table contains {proper noun}
                then T-Names = T-Names + Table
        endif
    endfor
endfor
return T-Names
```

- Synonym({word})

  This function identifies the synonym corresponding to the word. The synonym can be found in the topic-independent-semantic table.

Based on a number of semantic rules and the grammatical components in the question, the programming-steps generator generates instructions. Examples are provided in the following.

A Proper Noun

A grammatical component in the question can be a proper noun, which implies that it has a grammatical meaning of a proper noun. One set of semantic rules is that the programming-steps generator transforms the proper noun into instructions to select one or more topic-related tables, and then transforms other grammatical components in the question into instructions to select and to operate on data in the tables for answering the question.

Using the topic-dependent semantic table, the programming-steps generator first retrieves all tables where the proper noun is an attribute. Then, as shown in the topic-

dependent semantic table, all key attributes in those tables are identified, and each of them is matched to the proper noun. The table of any key attribute that matches the proper noun is selected for additional operation by the remaining grammatical components in the question.

In one example, the corresponding instructions are as follows:

```
for Table in Table-Of({proper noun})
do
    for Key in Keys-Of(Table)
    do
        x = (SELECT ...
            FROM Table
            WHERE Key MATCH {proper noun})
            // The above clause has the meaning of "where the key attribute
            // in the table matches the proper noun."
        if x is valid then done
        // if the SELECT function successfully identifies one or more attributes,
        // x is valid.
    endfor
endfor.
```

Common nouns

One grammatical component in the question can be a common noun. The programming-steps generator might transform the common noun into instructions to select a topic-related table, an attribute name, a synonym of an attribute name, the data under an attribute, or an algorithm.

If the noun denotes an attribute name or a synonym of an attribute name, again as shown by the topic-dependent semantic table, the programming-steps generator searches and identifies the attribute based on the noun. After all of the relevant attributes have been identified, data in them are retrieved for further processing by other parts of the question to generate an answer.

If the noun denotes the data under an attribute, the programming-steps generator identifies the data, with its corresponding attribute and table. The instructions generated can be, for example, (1) identifying each table in the function Tables-Of({noun}); (2) for each table identified, the function Attribute-Names({noun}, Table) returns the corresponding attributes containing the {noun} in that table; and (3) the remaining parts of the question operate on information under each attribute to generate the answer to the question. One set of instructions achieving such objectives is as follows:

```
for Table in Tables-Of({noun})
do
    ...
    for Attribute in Attribute-Names({noun}, Table)
    do
        SELECT ...
            FROM Table
            WHERE Attribute = {noun}
        ...
    endfor
    ...
endfor
```

The programming-steps generator might identify the algorithm corresponding to the noun; the algorithm is then applied to data selected by grammatical components in the question other than the common noun.

Non-Auxiliary Verbs

One grammatical component can be a non-auxiliary verb. It relates to one or more events or an action, which has a number of attributes; and it might have words with similar meaning. One approach is to identify the verbs with similar meaning. Then other components in the question identify data in the attributes of the identified verbs for answering the question.

A verb can be related to many different events. As an example, the verb is "nominate": one event can be President Bush being nominated to be the President, and another event can be President Clinton being nominated to be the President.

However, an event is related to a verb. The attributes of the event can have a subject-agent, which is the agent performing the event, such as the party nominating the president. Typically, the preceding noun phrase before the verb identifies the subject-agent. The event can have an object-agent if the verb is a transitive verb, which is the agent acted upon by the event, such as the president being nominated.

Each event has a duration that is between a starting and an ending time. For example, if the event is "walk," its duration starts with the sole of a foot changing its position from touching the ground to not touching the ground, and then ends with the sole back to touching the ground again.

Non-auxiliary verbs are grouped together in an event table, which is a topic-related table, with the topic being events. The following is an example of an event in the table:

```
CREATE TABLE EVENT (
    Verb_word           Character String NOT NULL,
                                // The verb that associates with the event
        Subject_Agent   Character String,  // Agent name performing the event
        Object_Agent    Character String,  // Agent name acted upon by the
                                           //event
        Start_Time      Time,              // Starting time of event
        End_Time        Time,              // Ending time of event
        Description     Character String,  // Describes the event
        KeyId           Integer,           // Unique number identifying the event
)
```

The subject-agent, object_agent etc. are attributes related to the verb_word, which is associated with an event.

There might be non-auxiliary verbs with similar meaning as the non-auxiliary verb in the question. These verbs can be identified by the synonym in the topic-independent semantic table. As an example, the verbs of breathe and inhale have similar meaning.

The programming-steps generator transforms the non-auxiliary verb in the question into one or more instructions, which select one or more verbs with their attributes in the event table. The one or more verbs have similar meaning as the non-auxiliary verb. Then other components in the question identify data in the attributes for answering the question. The selected verbs can be put into a temporary table or a view (a database terminology) as follows:

```
CREATE VIEW Verb_View({verb}) As
        // View is a logical table that is created only when it is needed.
        // All events matching {verb} are grouped from the event table
        // to form the view.
        SELECT * FROM EVENT
                // here * denotes all of the attributes
        WHERE Synonym({verb}) = Verb_word;
```

The attributes of the selected verbs are also identified. Then, the programming-steps generator generates additional instructions based on other components in the question to identify data in the selected attributes for answering the question.

Events might be related. Two events may form a sequential relationship, where one event follows another event, such as eat and drink. Two events may form a consequential relationship, such as braking and stopping, with the braking event causing the stopping event. Many small events may make up a big event, with the big event containing the small events; this leads to containment relationships. Also, events may be related because they involve the same subject-agent; and events may be related because they involve the same object-agent.

An event-relationship table describes relationships among events. It can have the following format:

```
CREATE TABLE EVENT_RELATIONSHIP (
        KeyId1          Integer,        // KeyId of an event
        KeyId2          Integer,        // KeyId of another event
        Relationship    Character String,
                //Relationship, such as sequential, consequential, containment etc.
)
```

Interrogative Pronouns

Based on the interrogative pronoun in the question, the programming-steps generator generates one or more instructions to select one or more attributes in one or more tables. Those tables have been selected by grammatical components in the question other than the interrogative pronoun. The function Attribute-Name({i-pronoun}, Table) generates the attribute name corresponding to the {i-pronoun}.

One way to generate a SQL-like instruction corresponding to the {i-pronoun} is to modify a SELECT clause:

SELECT Attribute-Name({i-pronoun}, Table) FROM Table

Determiners

Examples of a set of semantic rules on determiners are:

If the determiner is "a" or "an," select any result from the previous query.

If the determiner is "some," select more than one result from the previous query. If the previous query yields only one result, that result will be selected.

If the determiner is "all," select all result from the previous query.

If the determiner is "the," modify the following SELECT function with DISTINCT, as will be shown by examples below.

Auxiliary Verbs

An auxiliary verb together with either its immediate noun phrase or a non-auxiliary verb determine whether the answer should be singular or plural.

Adjectives

One grammatical component of the question can be an adjective. Based on the adjective, the programming-steps generator either identifies the value of an attribute, or identifies an algorithm. The grammatical components in the question other than the adjective have already selected one or more topic-related tables.

As shown by the topic-independent semantic table, the adjective may identify an attribute. The function Attribute-Names({adjective}, table) can retrieve the attribute in the table previously selected. The corresponding instruction can be:

```
for Attribute in Attribute-Names({adjective}, Table)
do
    SELECT ...
        FROM Table
        WHERE Attribute = {adjective}
        // or "Where the attribute in the table is equal to the adjective."
    ...
endfor
```

An adjective can refer to an algorithm, as identified by the topic-independent semantic table. Grammatical components in the question other than the component that is the adjective have selected one or more topic-related tables. As shown in the topic-independent semantic table, the adjective identifies one or more attributes in those tables. Then the algorithm operates on one or more data in those attributes.

Preposition

One grammatical component can be a preposition. A preposition can modify its previous noun phrase or verb, such as by operating on them through an algorithm identified in the topic-independent semantic table. Under some situations, with one or more tables selected by at least one grammatical component in the question other than the component that is the preposition, the algorithm identified operates on data or values in the one or more selected tables.

Under some other situations, for example, due to the prepositions 'of' and 'in', the programming-steps generator processes the grammatical component succeeding the preposition before the grammatical component preceding.

For another example, the preposition 'before' can modify the WHERE clause with a comparison on time:

{time of preceding event} < {time of succeeding event}

Programming-Steps Executor

The executor executes at least one set of instructions generated from one grammatical component to at least access data from the database to generate an answer for the question, if there is one.

In one embodiment, after the programming-steps generator generates a set of instructions, the programming-steps executor executes them. The set may be generated from one grammatical component. This process repeats until all sets are generated and executed to answer the question. For at least one set of instructions, the executor accesses data from one or more topic-related tables identified by the instructions. In another embodiment, all the instructions are generated; then the program executor runs the instructions, which include accessing data from one or more topic-related tables identified by the instructions, and processing those data for generating the answer to the natural-language question.

Example

The following shows examples of instructions automatically generated to answer grammatically-context-free questions.

1. Who is the first President?

```
for Table in each Tables-Of(President)
do
    for Attribute1 in Attribute-Names(President, Table)
    do
        for Attribute2 in Attribute-Names(first, Table)
        do
            res = (SELECT DISTINCT Attribute-Name(who, Table)
                FROM Table
                WHERE Attribute1 = "President"
                ORDER BY Attribute2 ASC)
```

```
                    if (res is not empty) return {first element of results}
                end for
            end for
        end for
5       return {error, no solution found}
```

As clearly shown in this example, the analysis starts with the noun phrase, *the first President*, and works towards the i-pronoun, *who*.

```
10      2.  What are the Bills of Right?
        answer = ""
        for Table in each Tables-Of("Bills of Right")
        do
            for Key in Keys-Of(Table)
15          do
                x = (SELECT Attribute-Name(what, Table) FROM Table
                        WHERE Key LIKE 'Bills of Right');
                answer = answer + x
            endfor
20      endfor
        if answer is not empty, return answer, otherwise return error.
```

As clearly shown in this example, the analysis starts with the noun phrase, *the bills of rights*, and work towards the i-pronoun, *what*.

25

Ambiguous Questions

The grammatical structure analyzer may decide that the natural-language question cannot be parsed into grammatical components based on the pre-defined context-free grammatical structure. For example, the grammatical components of the question cannot fit

into the pre-defined structure. Then the question is considered ambiguous, and an answer cannot be generated by the above method.

Ambiguity may be due to a number of reasons. For example, the question may contain words with non-unique grammatical meaning, the question may contain words not in the grammatical table, or the grammatical structure of the question is different from the pre-defined grammatical structure.

The grammatical structure analyzer can decide that a word can be of more than one grammatical meaning, such as it can be a noun and a verb. In one embodiment, the analyzer produces an answer for each meaning and ignores those meaning with no answer. In another embodiment, the analyzer asks the user to identify the correct grammatical meaning.

If the grammatical structure analyzer decides that the question contains one or more words not in the grammatical table, in one embodiment, the analyzer removes the un-recognized word and processes the remaining words in the question. In another embodiment, the analyzer asks the user for a different word. The analyzer might assume that the word is mis-spelled, and ask the user to correct it; the analyzer might replace the un-recognized word with a word in the grammatical table most similar to or with minimum number of different characters from the un-recognized word. The analyzer then presents the matched word to the user to ask if that is the right word. A list of matched words may be presented for the user to select.

Also, the answer generator can present suggestions to the user on ways to rephrase the original question based on the noun and the non-auxiliary verbs. It would then be up to the user to select the one he wants.

Questions Matching Engine

Another embodiment of the answer generator provides answers even to non-natural-language questions, and grammatically-context-dependent questions. In this embodiment, the database includes a questions table, which contains many questions, each with its corresponding answer. A question matching engine compares the question entered with questions in the database. An answer retriever retrieves the answer to the question in the database that matches the entered question. If no question in the database

matches the input question, the answer generator might use one of the approaches discussed in the ambiguous questions section to answer the question.